

1

2
3
4

5

6
7
8
9
10
11

12

13
14
15

1 BACKGROUND OF THE INVENTION

2 E-commerce protocols for bidding use commitments
3 enabling bids to be submitted in a way that the
4 commitment is secured until such a time that the bidder
5 wants it to be known. The type of commitment used
6 heretofore is herein referred to as a regular
7 commitment. It would be advantageous to employ a
8 commitment that has other properties, in accordance
9 with the present invention which has advantages of use
10 wherever a regular commitment is otherwise heretofore
11 used. It furthermore opens up additional applications
12 for commitments.

13 It is advantageous to discuss digital signatures, which
14 is a representative application for advantageous use of
15 the TCR-commitment. A digital signature is a
16 fundamental cryptographic primitive which, in the
17 digital setting, generally serves the same purpose as a
18 handwritten signature in the real world setting.
19 Namely, it binds, in an unforgeable, non-repudiable
20 manner, the identity of the signer to the information
21 being signed.

22 The mechanisms of applying and using public-key based
23 digital signature technology however, are quite
24 different than those used for handwritten signatures.
25 Firstly the digital signature itself is a mathematical
26 function of the message being signed and a private key

1 known only to the signer. Thus, unlike a handwritten
2 signatures, every digital signature employed by an
3 entity is very different from its other signatures,
4 since the underlying message being signed is likely to
5 be different. Verification of a digital signature,
6 therefore, is also very different from its handwritten
7 counterpart. Instead of examining the similarity
8 between the signature on a message and a sample
9 signature of the signer, the verifier applies a
10 mathematical function on the message, the signature and
11 a publicly known key of the signer. The publicly known
12 key is known to correspond to the signer's privately
13 held key. The system works as long as it is infeasible
14 to compute the private key from the publicly known key,
15 and without the private key it is infeasible to compute
16 the signature on any new message. Therefore, whereas
17 everyone is able to verify the signatures using the
18 public key, only the owner of the private key can
19 compute digital signatures.

20 The above is a description of how public key signatures
21 work in theory. However, to make such signatures useful
22 in practice, several additional steps, cryptographic
23 primitives and infrastructure are required. We now
24 describe some of these additional steps, primitives and
25 infrastructure since they have a direct bearing on this
26 invention.

1 From the description of the generation, signing and
2 verification process above, it should be clear that a
3 digital signature only binds the signature to the owner
4 of a private key corresponding to the public key used
5 in the verification process. Since private-public key
6 pairs are just mathematical objects which can be
7 generated by anyone, to serve any useful legal and/or
8 other purpose, there should also be a binding between
9 the public key and the real-world identity of the owner
10 of the corresponding private key. In practice this is
11 done by means of another digital object called a public
12 key certificate. The certificate (or chain of
13 certificates) is essentially a statement binding the
14 real-world identity of a signer with a public key,
15 together with a digital signature (or chain of digital
16 signatures) from a trusted entity vouching for the
17 veracity of the binding statement. The trusted entity
18 is referred to as a certification authority. Since the
19 certification authority's public key is well known,
20 this provides a way for another entity which trusts the
21 certification authority, to verify that indeed the
22 signer with a certain real-life identity is in
23 possession of the private key corresponding to the
24 public key. This other entity can verify the binding
25 between the signer's identity and public key by
26 verifying the certification authority's signature on
27 the binding object.

1 There is yet another important practical issue relating
2 to the use of digital signatures. In popular public key
3 signature schemes such as the RSA algorithm, the DSA
4 algorithm etc, once the public/private key for a signer
5 is chosen, the core signature algorithm can only be
6 applied to message blocks of a certain fixed size.
7 However, it is impractical for entities to have
8 multiple certified public keys for all possible message
9 block sizes. Also, computing signatures for very large
10 message-block sizes is very time consuming. In
11 practice, core signature schemes are used in
12 conjunction with another cryptographic primitive known
13 as a cryptographic hash function. A cryptographic hash
14 function is a publicly known function that takes in an
15 arbitrary sized message as an argument and produces a
16 fixed sized output. This output is said to be the
17 cryptographic hash of the message. The hash function
18 has a property of being collision resistant. A function
19 is collision resistant if it is infeasible to come up
20 with two different inputs upon which the function
21 produces the same output. Examples of cryptographic
22 hash functions include SHA-1 and MD5 which produce
23 160-bit and 128-bits outputs respectively.

24 With a collision resistant hash function, a signer can
25 sign any arbitrarily sized message with a single
26 private key by applying the signature algorithm only to
27 the cryptographic hash of the message. Similarly, a
28 verifier can verify a signature on a message by

1 applying the signature verification algorithm on the
2 cryptographic hash of the message. Since it is
3 infeasible for anybody (including the
4 commiter/signer/bidder, the verifier or any other third
5 party) to compute two different messages which have the
6 same cryptographic hash, with respect to unforgeability
7 and non-repudiation, a signature on the hash behaves
8 like a signature on the actual message.

9 In many cases, another cryptographic primitive called a
10 Target Collision Resistant (TCR) hash function can be
11 used in the place of collision resistant hash function
12 in the signature computation process for arbitrary
13 messages. TCR hash functions may be referred to as
14 universal one-way hash functions. Target Collision
15 Resistance can be viewed as a weaker requirement than
16 collision resistance, but is nevertheless considered to
17 be good enough for signatures. A target collision
18 resistant function is a publicly known function that
19 takes as input a fixed sized message and a fixed sized
20 key and produces an output of some fixed size. The
21 cryptographic property required of a TCR function T is
22 that given its description, it is infeasible for
23 someone to come up with any message M1, such that, when
24 the function T is computed with a randomly chosen key
25 K, it becomes feasible for that someone to compute some
26 other message M2, such that the value of T on M1 and K
27 coincides with the value of T on M2 and K. The point to
28 note in the definition of a TCR function is that the

1 key K is randomly chosen only after M1 has been chosen.
2 Indeed, it is quite possible that if K were to be
3 chosen first then it may be quite easy to produce a
4 collision, i.e., two messages M1 and M2 such that T on
5 K and M1 coincides with T on K and M2. Thus, the TCR
6 requirements of a function are less burdensome than the
7 requirements of collision resistance from a keyless
8 function which prohibits easy computation for any
9 collision whatsoever. Therefore, candidates for TCR
10 functions are expected to be easier to design than
11 candidates for collision resistant hash functions.

12 TCR functions are suitable for use in signatures in the
13 following way. In order to sign a message m, which in
14 the most general setting could have been chosen by
15 someone other than the commiter/signer/bidder, the
16 commiter/signer/bidder randomly picks a key k, computes
17 the TCR function on m and k and then signs the output
18 resulting from applying the TCR function together with
19 k using any available digital signature scheme. If the
20 available digital signature scheme is unforgeable, then
21 the scheme involving TCR function retains the property
22 that it is not feasible for anyone else to forge
23 signatures based on it . Hence they cannot be
24 repudiated . It should be noted that in the case of
25 TCR based signatures, the commiter/signer/bidder can
26 come up with two different messages which have the same
27 signature by choosing the messages after choosing the
28 key to the TCR function, whereas in the case of

1 collision-resistant hash functions, even the
2 commiter/signer/bidder cannot do so.

3 We now briefly describe another cryptographic primitive
4 known as a commitment scheme. Although, prior to the
5 present invention commitment schemes are unrelated to
6 digital signatures, they are described here, since they
7 will be shown to be relevant to the present invention
8 wherein the digital signatures is a digital commitment.
9 A commitment scheme is a two stage protocol involving a
10 committer and one or several verifiers. Informally, the
11 goal of a commitment protocol is to model some of the
12 properties of a sealed box or envelope in real-life.
13 The sealed box may, for example, include a committers
14 bid for an object. At the first stage of the protocol,
15 the verifiers are interested in having the
16 entity/committer irrevocably make a choice from a set
17 of options. The committer is amenable to making such a
18 choice but at this stage does not want the verifiers to
19 know what choice he has made. This situation often
20 arise in real-life in situations in which a party
21 (committer) wants to make a bid for an object without
22 disclosing the bid to anyone else. Everyone
23 participating in the bidding would require that each
24 party commit themselves to the bid so that no-one
25 should be able to change their bids at a later time. In
26 a more general setting, the commiter may be required to
27 commit to a data string instead of just a choice or a
28 bid.

1 The committer then follows the commitment protocol and
2 produces a digital message, known as a commitment
3 message. This commitment message has the property that
4 it forever binds the committer to a choice or bid, or
5 to the data string in the general setting without
6 disclosing any information about the data string. This
7 process is analogous, in the real-world to the
8 committer writing the data string on piece of paper and
9 putting the paper in a publicly displayed opaque
10 envelope or box and either sealing the envelope or
11 locking the box. At a later time, when all parties are
12 interested in publicizing the committed data string ,
13 the committer produces a commitment opening string
14 which together with the commitment message reveals the
15 data string that was originally committed. This in
16 real-life would correspond to the committer opening the
17 seal or lock to reveal the contents of the
18 envelope/box. This type of commitment is hereinafter
19 referred to as a regular-commitment. (The reason being
20 that this invention will describe a new cryptographic
21 primitive which will be termed a TCR-commitment, which
22 is very similar to a regular-commitment but
23 nevertheless has slightly weaker properties.)

24 It is well known in cryptographic literature that any
25 Regular Commitment Scheme is defined by the following
26 three functions.

1 Commitment function: This function takes a data
2 string to be committed together with secret
3 information known to the commiter to generate a
4 commitment message.

5 De-commitment function: This takes the secret
6 information known to the comiter (the one making
7 the commitment), the data string committed to and
8 the commitment message to produce an opening
9 string.

10 Commitment opening function: This function takes
11 an opening string together with a commitment
12 message to produce the data string committed to.

13 After having described public key signatures,
14 cryptograpic hash functions, TCR functions and regular
15 commitment schemes we now describe some of the
16 shortcomings of current digital signature technology
17 which the present invention addresses for use in a
18 digital commitment or digital signature commitment. A
19 digital signature commitment is used to commit a bidder
20 to a bid. The bid is herein also referred to as a
21 message and/or internal message which includes the bid
22 information.

23 One of the main drawbacks of current digital signature
24 technology which prevents its deployment in many
25 applications is performance. Public key signatures

1 using acceptable algorithms and key-lengths are very
2 compute-intensive to generate and/or to verify . For
3 example, even a high end workstation such as a 200 MHz
4 PowerPC 604e based RS/6000 can generate only 50 or so
5 1024 bit RSA signatures per second. For many
6 applications such a rate is unacceptably slow.

7 One such application is secure-multicast. In multicast,
8 a sender can send a single packet to a group of
9 recipients over a network. The network is responsible
10 for delivering the packet to all recipients in the
11 group. Multicast generally permits a far more
12 efficient usage of network bandwidth than the
13 alternative unicast approach in which the sender sends
14 the same packet individually to each of the recipients.
15 Now consider the problem of authenticating the sender
16 of a multicast packet. In the case of unicast, the
17 sender can append a message authentication code (MAC)
18 to the packet using a key it shares with the recipient.
19 In the case of multicast this does not work since the
20 MAC key would have to be known by all recipients and
21 therefore any recipient can masquerade as the sender.
22 The security exposure of using MACs for authentication
23 in multicast can be significant especially if the
24 multicast group is very large. An approach to solve
25 this problem from a security perspective requires the
26 sender to digitally sign each packet. Any recipient
27 would then be able to authenticate the packet without
28 having the ability to masquerade as the sender.

1 However, due to the poor performance of signature
2 schemes such an approach is not practical. Some
3 multicast applications require a packet rate exceeding
4 50 packets/s. Most applications which require less
5 throughput, nevertheless cannot devote a significant
6 fraction of the sender's computing resources to
7 calculating signatures. Also, if a sender is serving
8 multicast data to several multicast groups, the number
9 of public-key signatures that it can perform per second
10 per group is severely limited. Thus this solution is
11 not practically feasible.

12 Another possible solution is to relax the security
13 requirements for authentication. If a certain amount of
14 risk is acceptable, then one could use a fast but less
15 well-studied signature algorithm, such as those based
16 on Elliptic Curves. This approach would provide fast
17 yet secure authentication only to the degree to which
18 the underlying cryptographic assumptions are valid.
19 However, when these assumptions turn out to be invalid,
20 these schemes may be completely open to compromise.

21 Another approach to packet authentication when the
22 security requirement is relaxed is to use so-called
23 "asymmetric MACs". The basic idea is that one can use a
24 scheme in which multiple Message Authentication Codes
25 (MACs) are appended to each packet. This has the
26 property that the packet can be authenticated by each
27 receiver. It requires at least a threshold of

00110-9000500

1 malicious receivers to collude in order to masquerade
2 as a sender. However, the number of MAC's computed by
3 the sender and attached to the packet needs to be a
4 linear function of the number of colluders that the
5 scheme is supposed to be resilient against. Therefore,
6 even though this scheme may be useful in some
7 scenarios, e.g., when groups are small and problems of
8 collusion can be controlled, it does not work well in
9 scenarios where the multicast group is very large and
10 large collusions are likely to occur or difficult to
11 detect.

12 When reliability of transmission is not an issue, an
13 approach known as stream signing may be useful. It is
14 used to sign the multicast packets and yet provide a
15 relatively high level of security guarantees of
16 authentication and nonrepudiation associated with
17 digital signature/commitments. In this approach only
18 one regular signature is transmitted at the beginning
19 of a stream of packets. Each packet either contains a
20 cryptographic hash of the next packet in the stream, or
21 a 1-time public key using which the next packet can be
22 verified. . However, this approach cannot tolerate a
23 lost packet, since the information needed to
24 authenticate future packets is lost.

25 Given that IP-Multicast is generally an unreliable
26 protocol and/or many mulitcast applications such as
27 audio and video delivery are unlikely to require

1 reliability this solution may not be applicable. Also,
2 even if reliability is not an issue this solution
3 requires use of 1-time keys and signatures to be
4 embedded in each data packet. Such keys and signatures
5 are fairly large and can result in a substantial space
6 overhead within each packet which may be unacceptably
7 large. Sometimes, the stream signing overhead can be
8 substantially reduced at the cost of introducing a
9 delay at the sender side and grouping several packets
10 together so as to distribute the overhead into multiple
11 packets. However, delaying and grouping of sender's
12 packets cannot be done in peer-to-peer interactive
13 multicast applications such as distributed simulations
14 and gaming which are important multicast usage
15 scenarios.

16 Another approach is useful when a sender is allowed to
17 delay and group together several consecutive packets.
18 In this approach, the sender uses a collision resistant
19 hash function to form an authentication tree from
20 packets collected during a time-interval and signs the
21 root of the authentication tree. Each packet is
22 augmented by the signature on the root and ancillary
23 information. The ancillary information includes hashes
24 of neighboring nodes on the logarithmically long path
25 from the message to the root of the authentication
26 tree. The collected and augmented packets are then
27 sent. This allows each packet to be individually
28 verified at the receiving end.

1 While this approach is quite effective in client-server
2 scenarios where the server is dedicated to serving a
3 small number of multicast flows, each with reasonably
4 smooth flow rates and strictly enforced bounds on
5 processor loading, it suffer from several practical
6 drawbacks. Firstly, delaying and grouping of sender's
7 packets is not always possible for many important
8 multicast applications. Secondly, there is still a
9 problem of serving multiple multicast flows. This is
10 best illustrated by the following example. Suppose a
11 server only has enough cycles to perform 10 public key
12 operations per second. Using this approach, such a
13 server could potentially serve hundreds of
14 authenticated packets per second to a single flow with
15 only a minor delay, say a tenth of a second. However,
16 if the same server was required to send only 50
17 different flows, each with a packet rate of only 1
18 packet/second (such as serving multiple hand held
19 devices) for a total of 50 packets/second throughput,
20 this will not be possible unless the same signing key
21 and authentication tree data structure is shared
22 across different flows, or an unacceptably long delay
23 of 5 seconds is imposed on each flow. Sharing between
24 flows puts unreasonable constraints on the software
25 architecture at the server side. This restrict the
26 choice of authentication mechanisms and expose privacy
27 issues regarding combining information from different
28 flows. A third problem with the scheme is that the size

1 of the authentication information tacked on to each
2 message is not fixed. The size depends of the
3 short-term packet rate which in many applications is
4 likely highly irregular and bursty. During bursty
5 periods, the packets will be larger. This can cause
6 undesirable side-effects such as increased packet loss
7 due to fragmentation. A fourth problem with the scheme
8 is that it provides no mechanism to smooth out bursty
9 processor loads. In any real system there are periods
10 when the processor has enough free time to calculate 50
11 signatures/s. However, there are also times when the
12 processor barely has time to calculate 1-2
13 signatures/s. With the tree approach, there is no way
14 to leverage the idle time of the CPU to help during the
15 time when it is highly loaded. Performance will
16 seriously degrade when the CPU gets loaded.

17 SUMMARY OF THE INVENTION

18 Thus the present invention provides what is herein
19 referred to as a TCR-commitment. The TCR commitment
20 has properties differing from the properties of a
21 regular commitment which makes it useful in a variety
22 of applications.

23 In one aspect, the present invention addresses the
24 problem caused by the slow speed of public key
25 commitment/signature algorithms. It solves the problem

1 of packet authentication for multicast and other
2 scenarios requiring fast, compact digital
3 signature/commitment for E-commerce protocols.

4 Another aspect of the present invention provides a
5 relatively high level of security guarantees required
6 for packet authentication in a way that can handle
7 multiple independent flows, produces authentication
8 fields of fixed size, works in the fully unreliable
9 setting, does not require any packet delays and has the
10 additional property of being able to withstand and
11 smooth over irregular processor loading and bursty
12 packet output rate.

13 An aspect of the present invention uses a hybrid
14 approach consisting of the commiter/signer/bidder
15 creating a certificate for the public key of an
16 efficient k-time signature scheme using a regular
17 signature key. The commiter/signer/bidder then signing
18 up to k messages with the private key corresponding to
19 k-time public key. The time consumed to compute a
20 public key signature is amortized over k signatures.

21 These and other aspects are provided in a commitment
22 scheme wherein a TCR-commitment or other commitment is
23 employed. Other objects and a better understanding of
24 the invention may be realized by referring to the
25 detailed description.

1 BRIEF DESCRIPTION OF THE DRAWINGS

2 These and other objects, features, and advantages of
3 the present invention will become apparent upon further
4 consideration of the following detailed description of
5 the invention when read in conjunction with the drawing
6 figures, in which:

7 Fig. 1 shows an example of a process for generating a
8 certified 4-time public key from 4 1-time public keys
9 using an authentication tree based on the collision
10 resistant hash function SHA-1;

11 Fig. 2 shows an example of a process of signing a
12 message with the 3rd use of a 4-time key and the
13 process of creating ancillary information;

14 Fig. 3 shows an example of a process of verifying a
15 signed message consisting of 3'rd signature from a
16 4-time public key and ancillary information;

17 Fig. 4 shows a high level description of an example
18 generation process of a 36-time key pair embodiment;

19 Fig. 5 shows the construction of the i'th one-time
20 signature key-pair with embedded enhancing key
21 commitment, as part of the construction of the 36-time
22 key pair in accordance with the present invention;

1 Fig. 6 shows the construction of the 36-time public key
2 from the 36 1-time public keys with embedded
3 commitments, using a TCR hash tree construction in
4 accordance with the present invention;

5 Fig. 7 shows the signature generation process on a
6 message using the i'th key in the 36-time signature
7 scheme in accordance with the present invention;

8 Fig. 8 shows the process of verification of the i'th
9 signed message in the 36-time signature scheme in
10 accordance with the present invention;

11 Fig. 9 shows an example of how a TCR commitment
12 function is built in accordance with the present
13 invention;

14 Fig. 10 shows an example of how a TCR De-commitment
15 function is built in accordance with the present
16 invention; and

17 Fig. 11 shows an example of how a TCR open function is
18 built in accordance with the present invention.

19 **DETAILED DESCRIPTION OF THE INVENTION**

20 Commitments have been used in several cryptographic
21 protocols including those dealing with proofs of
22 knowledge and bidding protocols. In some situations and

1 applications, one only requires a weaker property. In
2 these situations the commiter/signer/bidder may later
3 be able to change the committed value/bid with a
4 feasible probability. However, no-one else should be
5 able to create a different opening of the
6 commiter/signer/bidder's bid. The present invention of
7 TCR commitments, which is more efficient than
8 heretofore regular commitments addresses and improves
9 these protocols. The TCR-commitment is useful for many
10 other applications, E-commerce, etc.

11 Thus the present invention provides methods, apparatus
12 and computer articles which solve problems caused by
13 the slow speed of public key signature algorithms
14 without many of the drawbacks of other schemes.
15 Although the following discussion describes ways to
16 solve problems of packet authentication for multicast,
17 it should be obvious to those with ordinary skill in
18 the art that the same concepts and/or approach have
19 applications to many other scenarios requiring fast,
20 compact digital signature/commitments. The methods,
21 apparatus and computer articles are herein said to
22 employ a 'commitment based signature scheme' in
23 accordance with the present invention. More
24 particularly these employ a 'TCR commitment based
25 signature scheme'. In what follows the term message is
26 used to refer to an internal message which contains the
27 information being signed.

1 TCR commitments form a new type of cryptographic
2 primitive disclosed in this invention. Analogous to
3 regular commitment schemes we now define the concept of
4 a target collision resistant commitment scheme or a TCR
5 commitment scheme. A TCR commitment scheme consists of
6 the following three functions:

7 1. TCR Commitment function: This function takes a
8 data string to be TCR committed together with
9 secret information known to the commiter to
10 generate a TCR commitment message.

11 2. TCR De-commitment function: This takes the
12 secret information known to the TCR
13 comiter/signer/bidder, the data string TCR
14 committed to and the TCR commitment message to
15 produce a TCR opening string.

16 3. TCR Commitment opening function: This function
17 takes a TCR opening string together with a TCR
18 commitment message to produce the data string TCR
19 committed to.

20 The properties of the TCR commitment scheme are:

21 1) TCR commitment string gives no information as
22 to the value of the data string committed.

1 2) With high probability, it is computationally
2 infeasible for anyone who does not have influence
3 in the TCR commitment generation process to
4 create a different opening of the commitment given
5 knowledge only of a TCR commitment message and a
6 corresponding TCR opening string. Generally a
7 high probability is a probability of $1-(10^{-5})$ or
8 much greater. However each application decides
9 what is considered a high probability for its
10 requirements.

11 The present invention provides a level of security
12 appropriate for packet authentication using a TCR
13 commitment. Its performance is generally efficient in
14 speed and size, can handle multiple independent flows,
15 produces authentication fields of fixed size, works in
16 a fully unreliable setting (with additional small
17 overhead), and does not require any packet delays. It
18 has the additional property of being able to withstand
19 and smooth over irregular processor loading and bursty
20 packet output rate. In some ways the invention mimics
21 the public key signature per packet approach which is
22 an ideal solution to this problem when the speed of
23 public key signature algorithms is not an issue.

24 The present invention adopts a hybrid approach based on
25 the following realization. Although public key
26 signature schemes are slow, a private key can be used
27 an unlimited number of times. There exist well known

1 secure 1-time, or k-time for some constant k, signature
2 schemes in which key generation, signature generation
3 and verification is extremely fast. However, the
4 public-private key pair for such schemes can be used at
5 most once, or k times respectively. Several
6 1-time/k-time schemes are known to those familiar with
7 the art.

8 Therefore the use of the hybrid approach in which a
9 commiter/signer/bidder creates a certificate for the
10 public key of an efficient k-time signature scheme
11 using his regular signature key, and then signs upto k
12 messages with the private key corresponding to the
13 k-time public key results in a very fast hybrid
14 signature scheme. This is because the time consumed to
15 compute a regular public key signature to create the
16 certificate is amortized over k message signatures.
17 This may be viewed as a direct generalization of an
18 off-line/on-line signature scheme in which the off-line
19 expensive computation uses a regular signature on a
20 1-time public key, and the on-line fast computation
21 uses a 1-time signature.

22 Using this hybrid approach, packet authentication may
23 be done as follows. Either the sender, or an entity
24 trusted by the sender, runs a background key generation
25 process. This process creates a buffer of k-time
26 public/private key pairs and creates certificates for
27 each of the k-time public keys using a regular

1 signature scheme. In an example embodiment this is done
2 by creating k, 1-time public/private key pairs and
3 combining them to form a a k-time public/private key
4 pair. The k-time public key is determined by computing
5 a hash tree of k 1-time public keys and setting the
6 root of the tree to be the k-time public key. The
7 certificate for the k-time public key is the signature
8 of the sender on the root of the hash tree. This may
9 use the sender's regular signature scheme.

10 An example of such an approach is illustrated in Figure
11 1 for k=4. Figure 1 shows how a 4-time public key and
12 certificate is built from 4 1-time public keys PK1
13 (111), PK2 (112), PK3 (113), and PK4 (114), the SHA-1
14 collision resistant hash function and a regular
15 signature scheme . The first step is to form a
16 authentication tree from the 4 1-time keys PK1, PK2,
17 PK3 and PK4 (111-114) by putting them at leaves of a
18 binary tree (100). At the next higher level of the
19 binary tree there are 2 nodes PK12 (121) and PK34
20 (122), where the value of PK12 (121) is derived from
21 PK1 (111) and PK2 (112) by applying the SHA-1 hash
22 function to the concatenation of PK1 and PK2, and the
23 value of PK34 (122) is similarly derived from PK3 (113)
24 and PK4 (114). At the root of this tree is the node
25 PK1234 (130) . This node is the 4-time public key which
26 is derived by applying the SHA-1 hash function to the
27 concatenation of the values of the 2 nodes at the lower

1 level in the tree, i.e., PK12 (121) and PK34 (122) .
2 The root PK1234 (130) together with a regular signature
3 on the root (140) form the certificate (150) for the
4 4-time public key PK1234 (130) .

5 An example of an embodiment of the process of computing
6 signatures on messages using this hybrid scheme uses
7 the following steps. Consider a time when the
8 commiter/signer/bidder is working with a particular
9 k-time key pair and its corresponding certificate.
10 Assume that the commiter/signer/bidder has already used
11 up (i-1) of the k uses of the k-time key. This is after
12 the commiter/signer/bidder has already performed (i-1)
13 signatures using (i-1) private keys which constitute a
14 subset of the k-time private key in the generation
15 scheme described above. To sign a given next message m,
16 the commiter/signer/bidder computes a collision
17 resistant hash h of the next message m, and signs h
18 with the i'th use of the k-time private key to form a
19 signature s. This corresponds to the i'th 1-time key
20 within the k-time key in our example. Now the
21 commiter/signer/bidder appends signature s with some
22 ancillary information. In this example this is the
23 i'th public key and hashes of all neighbors on the path
24 from the i'th public key to the root of the hash tree
25 formed during the construction of the k-time public
26 key. The purpose of this ancillary information is to
27 permit a verifier to verify from the k-time public key,

1 the i'th use of the corresponding private key. The
 2 signature s, the ancillary information and the
 3 certificate together provides enough information for a
 4 verifier to verify the signature on the message. Since
 5 the same certificate is to be used at the verifying end
 6 for all k uses of the key, depending on the amount of
 7 unreliability, in some embodiments the certificate is
 8 not sent for each signed message to the verifier. It is
 9 sent as a single separate packet or multiple packets.
 10 Alternatively, the certificate is embedded within each
 11 signed message packet or into a fraction of the signed
 12 message packets. The signature s and ancillary
 13 information is sent along with each message in order to
 14 verify the signature on the message.

15 In this embodiment, once a k-time key is fully used up,
 16 i.e., k messages have been signed using the key, the
 17 commiter/signer/bidder proceeds to the next available
 18 k-time key and its corresponding certificate from its
 19 buffer of other k-time keys and certificates. Figure 2
 20 illustrates an example of a signing process for a
 21 message m based on the 3'rd use of the 4-time key
 22 PK1234 (130) derived in Figure 1, and the SHA-1
 23 collision resistant hash function. To sign m, first
 24 $h = \text{SHA-1}(m)$ is computed (210). To sign m in a third use
 25 of PK1234 (130), use the secret key, SK3, corresponding
 26 to the 1-time public key PK3 (113) to generate a
 27 1-time signature s on h (220). Compute the ancillary

1 information (230). In this case the ancillary
2 information is PK3 (113), PK4 (114) and PK12 (121) .

3 An example of the process of verifying signed messages
4 using this scheme has the following steps. The verifier
5 has access to the certificate for the k-time key being
6 used to sign all messages. It can obtain the
7 certificate in many ways as outlined earlier. The
8 certificate may be a separate packet or embedded within
9 some or all message packets. In one embodiment the
10 certificate receiver/verifier builds a cache of
11 received certificates. After verifying the
12 certificate, the verifier has a verified k-time public
13 key. In this example this is the root of the hash tree
14 formed from k 1-time public keys. Along with each
15 message m, the receiver/verifier also receives a
16 signature, s, and ancillary information corresponding
17 to some i'th use to the k-time key on the message. The
18 receiver computes h, the collision resistant hash of m
19 and verifies the signature s on h, with respect to the
20 i'th public key. It then uses the ancillary information
21 fto authenticate the i'th public key with respect to
22 the k-time public key. In this example this is done by
23 verifying that the provided ancillary information is
24 consistent with a hash tree with the i'th public key as
25 a leaf and the k-time public key as a root.

1 Figure 3 shows an example of a process for verifying a
 2 signed message m as created in Figure 2. The signed
 3 message is assumed to consist of the message m, the
 4 signature s and ancillary information PK3, PK4 and PK12
 5 (310). The SHA-1 hash h of the message m is created
 6 (320). The supplied signature s on h is verified
 7 against the alleged public key PK3 (330). If
 8 verification produces a match, then the signature is
 9 valid provided that the supplied public PK3 is the same
 10 as that used in the derivation of the authentication
 11 tree in Figure 1. The validity of the supplied PK3 is
 12 checked as follows. First the supplied values of PK3
 13 and PK4 are hashed using SHA-1 to derive the alleged
 14 value of the interior node PK34 within the
 15 authentication tree shown in Figure 1 (340). The value
 16 of the supplied node PK12 and the computed value of
 17 PK34 are hashed using SHA1 to derive the alleged value
 18 of the root node PK1234 of the hash tree (350). This
 19 derived value is compared against the signed value of
 20 PK1234 in the verified certificate for PK1234 (130) and
 21 if it matches then the signature on m is valid,
 22 otherwise it is invalid (360).

23 This method solves the problem related to signing
 24 speed. This is because the cost of a single expensive
 25 public-key signature is amortized over k packets. The
 26 per-packet cost associated with 1-time/k-time signature
 27 is fairly inexpensive. This method also handles

1 multiple flows, bursty traffic and bursty processor
2 load very well. This is because the buffers of
3 certified k-time keys can be precomputed and filled up
4 during periods of low CPU usage and slow traffic, to
5 tide over periods of high CPU usage and high traffic.

6 However, this approach suffers from a drawback which
7 may make it impractical for many uses. This drawback is
8 the size overhead of state-of-the-art one-time/k-time
9 signature schemes, together with the overhead of the
10 ancillary information. One-time/k-time schemes have
11 either very large signatures or public keys or both.

12 This overhead can easily amount to half a kilobyte per
13 packet or more. This is unacceptable for many
14 applications. For example, many UDP/IP-Multicast
15 applications limit the packet size to a few hundred
16 bytes to prevent additional packet loss due to
17 fragmentation. Although 1-time/k-time schemes
18 generally have some tradeoff between size and speed the
19 tradeoff is overwhelmingly biased towards speed, such
20 that a small reduction in size normally results in a
21 very large decrease in speed.

22 Other size overheads in the scheme described include
23 the size of the certificate for the k-time public key
24 and the size of the ancillary information attached to
25 each signature.

1 An improvement to the method of the present invention
2 makes it possible to dramatically cut down on the size
3 overhead associated with the signature scheme.
4 Whereas, the motivation for these ideas is to cut down
5 the size of signatures and ancillary information, this
6 size reduction results in comparable speed improvements
7 on the underlying 1-time/k-time signature schemes as
8 well. In embodiments in which size of signatures is
9 not an issue, and given the heavy bias in the
10 speed/size tradeoff for 1-time/k-time schemes, these
11 techniques can also be used to create
12 disproportionately much faster 1-time/k-time signature
13 schemes with signature sizes similar to schemes which
14 do not use these techniques. For example, those
15 skilled in the art will recognize that one could start
16 with a larger sized signature scheme which is much
17 faster and then use the size reduction techniques
18 described here to cut down the size of the signatures
19 back to a more useable size. This is done while
20 retaining or even improving upon the speed.

21 Many 1-time/k-time signature scheme have a signature
22 size and speed which is proportional to the number of
23 bits of the message being signed. Typically, to deal
24 with arbitrary sized messages, the actual value that is
25 signed is a collision resistant hash of the message.
26 However, the signing operation can also be performed
27 using TCR hash functions instead of collision
28 resistant hash functions. An advantage with TCR hash

1 functions is that their output size need not be as
2 large as the output of a collision resistant hash
3 function. This is because several types of attacks
4 based on the size of the output are not applicable to
5 TCR functions. In practice this means that the output
6 size of a TCR hash function can be only one half as
7 large as a comparable collision resistant function.
8 This may not provide significant size advantage for
9 1-time/k-time signature schemes. This is due to
10 security reasons which may require a signature on both
11 the output of the TCR function and the randomly chosen
12 TCR key. This key is part of the TCR function
13 calculation after the message to be signed has been
14 fixed. A further size reduction results by using a
15 scheme in which this TCR key does not have to be signed
16 using the one-time/k-time signature scheme. To retain
17 security, the TCR key should satisfy two properties.
18 The TCR key should be signed somehow by the
19 commiter/signer/bidder and this key should not be
20 disclosed before the message is fixed, otherwise, an
21 adversary that knows the key beforehand can influence
22 the message in a way that can break the signature
23 scheme.

24 A further improvement to the present invention is
25 provided based on the following consideration. For
26 any signature scheme, at the time of generating a
27 public-private key pair one can generate other digital
28 strings which are herein referred to as enhancing keys,

1 include regular-commitments to the enhancing keys as
2 part of the public key and retain the enhancing keys as
3 part of the private key. When creating the certificate
4 of the public-key, the certificate commiter/ signer/
5 bidder directly or indirectly be signs the
6 regular-commitments to all these enhancing keys. At
7 the time of performing a signature with the private
8 key, the commiter/signer/bidder uses one or more of
9 the enhancing keys to realize additional efficiencies
10 in the signature process and may open the commitments
11 to the some/all enhancing keys used in the process.

12 An example embodiment of the present invention using
13 these regular commitments performs the following steps
14 in a commitment based signature scheme. During the
15 generation of each k-time public-private key pair, the
16 commiter/signer/bidder chooses at random all the k TCR
17 hash keys which are later used in the TCR hash
18 calculation on the k messages that are signed by the
19 k-time key being generated. These are the enhancing
20 keys for this embodiment . The commiter/signer/bidder
21 includes regular-committments to these keys as part of
22 the k-time public key. Thus, when creating the
23 certificate for the k-time public-key the
24 commiter/signer/bidder directly or indirectly signs
25 the regular-commitments to all these enhancing keys.
26 Anyone who obtains this certificate or the k-time
27 public key inside therefore gets no knowledge of the
28 TCR-keys. This is so even though by signing the

1 commitment to the enhancing keys the
2 commiter/signer/bidder has in effect signed on to
3 ownership of these keys. Now when the
4 commiter/signer/bidder is asked to sign an i'th
5 message with the k-time public key, the
6 commiter/signer/bidder creates a signature that
7 includes an i'th instance of a k-time signature on
8 only the TCR hash of the message computed using the
9 i-th enhancing key and a commitment opening string to
10 the i-th enhancing key. . The signature verifier is
11 given the message and an i'th instance of a k-time
12 signature together with the opening string to the
13 regular-commitment to the TCR hash key. The verifier
14 derives the TCR hash key from the opening string and
15 uses it to compute the TCR hash of the message,
16 verifies the i'th instance of the k-time signature on
17 this hash and also verifies that the commitment opening
18 string is consistent with the signed regular-commitment
19 made in the certificate for the k-time public key.

20 In effect the i-th TCR key, which is the i'th
21 enhancing key becomes publicly known only after the
22 i-th message has been fixed and signed. For everyone
23 else other than the commiter/signer/bidder the process
24 is the same as if the commiter/signer/bidder had
25 actually chosen the TCR key at random after the message
26 was fixed. Thus the security or the semantics of the
27 signature scheme are not compromised. Moreover, the
28 size overhead of the commitment opening string is small

1 when compared to the size savings in applying a
2 typical 1-time/k-time signature only on the TCR hash of
3 the message. The TCR hash is usually only half as
4 large as the collision resistant hash of the message
5 which in itself yields a factor of 2 reduction in size
6 of the signature. Another benefit of this size
7 reduction is a factor of 2 improvement of the speed of
8 the generation, signature and verification times
9 1-time/k-time signature scheme.

10 Many k-time signature schemes are constructed from k
11 independent 1-time schemes, as was done in our example.
12 When such a k-time scheme is employed in conjunction
13 with the other concepts of the present invention,
14 another optimization results. This generally reduces
15 the size of the public key of the resulting k-time
16 scheme, the size of the certificate and the size of
17 ancillary information that is added along with each
18 1-time signature which authenticates the particular
19 i'th use of the k-time scheme. This is done as follows.
20 Following the previous example and using the new
21 1-time public key scheme with commitments, a k-time
22 public key is formed by computing a target collision
23 resistant hash function rather than the using a
24 collision resistant hash function based on the
25 authentication tree of k 1-time public keys as in
26 Figure 1. As an optimization, the tree is formed based
27 on the TCR hash of these k public keys. This tree is
28 herein referred to as a TCR tree. The TCR hash,

1 together with the keys used to compute the TCR tree
2 based TCR hash becomes the k-time public key.

3 This approach offers a big advantage in that verifying
4 that a node belongs to a leaf in a TCR tree, requires
5 much less ancillary information than the case of a
6 authentication tree based on a collision resistant hash
7 function for the same number of leaves. However, use of
8 a TCR function in creating a k-time public key weakens
9 the regular-commitment scheme embedded in the 1-time
10 public key. The commitment when used with a TCR tree is
11 herein referred to as being a "TCR-commitment". A
12 "TCR-commitment" produces an enhanced 'commitment based
13 signature scheme.' A TCR commitment is a type of
14 commitment in which it is possible for a committer to
15 later alter the committed value but has no incentive
16 to do so. However, the receiver of the TCR commitment
17 gets no information from the TCR-commitment, and also
18 cannot create a different consistent opening of the
19 TCR-commitment once the committer has opened the
20 TCR-commitment. This weakening therefore has little or
21 no impact on the security of the signature scheme but
22 results in a generally significant performance
23 improvement.

24 Referring back to the design of the k-time public key
25 using a TCR hash tree, which utilizes the invention of
26 a TCR-commitment, the k-time public key is further
27 signed using a regular signature to form the

1 certificate. Another optimization that is employed in
2 some embodiments of the present invention is the use of
3 a padding scheme known to those skilled in the art. This
4 scheme embeds the k-time public key inside the regular
5 signature itself for many popular public key signature
6 schemes and thus saves on the overhead of transmitting
7 both the k-time public key and a regular signature on
8 it.

9 An example embodiment of the present invention is
10 based on a specific 36-time signature scheme and
11 1024-bit RSA as the base signature algorithm. These
12 presently provide what is considered to be adequate
13 levels of security given the state of current knowledge
14 and technology. The embodiment provides a level of
15 security of the order of $2^{(80)}$ operations which is
16 considered adequate. By adjusting the input-output
17 sizes of the primitives described below, the scheme
18 generalizes to provide higher and/or lower levels of
19 security as required by a particular application.

20 The scheme is based on cryptographic primitives, each
21 of which offers around 80 bits of security. Such
22 primitives can be specifically designed or derived from
23 other known primitives. In what follows the primitive
24 is described in terms of its cryptographic properties.
25 A specific instance of that primitive is provided for
26 the example embodiment.

1 $G(K,x)$ is the first 80 bits of output of
 2 $\text{SHA-1Compress}(\text{Pad80to160}(K), \text{Pad80to512}(x))$

 3 Let $T_1(K,x)$ denote a target collision resistant keyed
 4 hash function family taking an 80 bit key K and a 480
 5 bit input x , producing an 80 bit output. The example
 6 embodiment for T_1 is as follows: Define $\text{Pad480to512}(x)$
 7 to be a function which pads a 480 bit input to 512 bit
 8 by adding trailing 0 bits. An advantageous embodiment
 9 of $T(K,x)$ is the first 80-bits of output of
 10 $\text{SHA-1Compress}(\text{Pad80to160}(K), \text{Pad480to512}(x))$.

 11 Let $T_2(K,x)$ denote a target collision resistant keyed
 12 hash function family taking an 80 bit key K , a 160 bit
 13 input x , and producing a 80 bit output. Let
 14 $\text{Pad160to512}(x)$ be defined to be a 160-bit to 512 bit
 15 function which pads its input to 512 bits by adding
 16 trailing 0's. In the example embodiment, $T_2(K,x)$ is
 17 the first 80 bits of output of
 18 $\text{SHA-1Compress}(\text{Pad80to160}(K), \text{Pad160to512}(x))$

 19 An example generation of the 36-time key is as follows.
 20 It is essentially derived from 36 independent random
 21 1-time public keys hashed down using TCR hash functions
 22 in a tree construction known to those skilled in the
 23 art. The 36-time private keys are some of the
 24 intermediate values used in the construction.

1 The example 36-time key generation outlined in Figure
2 4, is done as follows: Pick three 80-bit keys g_1 , g_2
3 and g_3 uniformly at random (400). These keys are
4 used throughout the construction of this 36-use public
5 key. Throughout the construction, $g(x) = G(g_1, x)$ is
6 used as an 80 bit to 80 bit one-way function (410). Now
7 for each i , (1 to 36) compute the i 'th key pair (420).

8 The details of computing the i 'th key pair are given in
9 Figure 5 and consists of the following steps. Pick T_i ,
10 an 80 bit key uniformly at random (500). T_i is used
11 as the key to the TCR function when the i 'th key is
12 used to sign a message. T_i 's therefore are the
13 enhancing keys in the invention. Pick 23 random 80 bit
14 values, $r_{\{i,1\}}, \dots, r_{\{i,23\}}$ (510). From these compute
15 $f_{\{i,1\}}, \dots, f_{\{i,23\}}$ as follows:

16 For each $j = 1, \dots, 22$, compute,

17 $f_{\{i,j\}} = g^{\{15\}}(r_{\{i,j\}})$, and

18 $f_{\{i,23\}} = g(r_{\{i,23\}})$ (520)

19 The i 'th public key which includes in it a "commitment"
20 for the TCR key T_i used for signing the i 'th message
21 is defined as ,

22 $PK_i = H(T_i \mid f_{\{i,1\}} \mid \dots \mid f_{\{i,23\}})$,

1 where "|" denotes concatenation (530). In this
2 construction the application of H results in a
3 commitment to T_i .

4 The public keys for each use of the 36-time signature
5 scheme are $PK_1, PK_2, \dots, PK_{36}$. These are combined
6 (refer back to Figure 4) to form PK36 the 36-time
7 public key (430) with details given below and in the
8 description of Figure 5.

9 The public key of the 36-time signature scheme is a
10 tree based TCR hash of the 36 public one-time signature
11 keys PK_1, \dots, PK_{36} , together with the keys used to
12 create the TCR hash. This public key is created as
13 follows as is illustrated in Figure 6.

14 Consider a tree in which the leaves are the 36 public
15 keys PK_1, \dots, PK_{36} each of which is 160 bits long
16 (610). The next higher level in the tree has 36 nodes,
17 L_1, \dots, L_{36} , one for each public key. The value of
18 node L_i is just $T2(g_2, PK_i)$, which is the TCR hash
19 of PK_i , using the family T2 under key g_2 (620). The
20 next higher level in the tree has 6 nodes M_1, \dots, M_6 .
21 Thus, each node M_i has 6 children,
22 $L_{(6(i-1)+1)}, \dots, L_{(6i)}$ and its value is computed as
23 $M_i = T1(g_3, L_{(6(i-1)+1)} \mid \dots \mid L_{(6i)})$ (630). These
24 6 nodes M_1, \dots, M_6 together with g_1, g_2, g_3 ,

1 constitute the 36-time public key PK36 (640). Since
2 each M_i is an 80-bit value, therefore the 36-time
3 signature scheme has a public key of size 720 bits.
4 Also by applying a Ttree-based TCR hash indirectly on
5 the commitments (T_i 's) made within individual 1-time
6 public keys, the individual commitments are
7 transformed into a TCR-commitment on all the T_i 's.

8 Now Consider Certificate Generation for the 36-time Key
9 PK. The certificate for the 720-bit, 36-time key
10 public key PK36 is created by applying the 1024-bit RSA
11 signature algorithm with PK36 as embedded data as
12 known to those familiar with the art. . This results in
13 a 1024 bit value which not only serves as a signature
14 on PK36, but also enables the process of verification
15 of the 1024 bit certificate produces PK36. Thus PK36
16 never needs to be sent explicitly with the certificate.
17 Thus the certificate is of size 1024 bits or 128 bytes.

18 Now consider generation of the signature for the
19 example embodiment shown in Figure 7. To sign a
20 message m (710) with the i 'th key of the 36-time
21 signature scheme, the commiter/signer/bidder does the
22 following:

23 Compute $h = T2(T_i, H(m))$. This is an 80 bit TCR
24 hash of m (720).

1 Let h_1, \dots, h_{80} denote the bits of h . Group these
2 bits into 20 groups of 4 consecutive bits (aka
3 nibbles). The value of each nibble is a number from 0
4 to 15. Let n_1, \dots, n_{20} denote these numbers (730).

5 For $1 \leq j \leq 20$, compute:

6 $Y_j = g^{\{15-n_j\}}(r_{\{i,j\}})$ (740),

7 where $r_{\{i,j\}}$ are the random numbers chosen during key
8 generation.

9 Compute $N = n_1 + n_2 + \dots + n_{20}$,

10 where N is a number between 0 and 300, and therefore
11 fits in 9 bits. Let n_{21} denote the value of the least
12 significant 4 bits of N , n_{22} denote the value of the
13 next 4 significant bits of N and n_{23} denote the value
14 of the most significant 9'th bit of N (750).

15 Form $Y_{21} = g^{\{n_{21}\}}(r_{\{i,21\}})$,

16 form $Y_{22} = g^{\{n_{22}\}}(r_{\{i,22\}})$, and

17 form $Y_{23} = g^{\{n_{23}\}}(r_{\{i,23\}})$. (760)

18 The signature then includes T_i, Y_1, \dots, Y_{23} together
19 with the values of the 5 other L nodes corresponding

1 to the children of $M_{\{(i+5)/6\}}$, excluding L_i , which
2 are the siblings of L_i , in the key generation process.
3 Let $Sib(L_i)$ denote these sibling nodes. Therefore the
4 signature includes $T_i, Y_1, \dots, Y_{23}, Sib(L_i)$. Note
5 that T_i represents the opening of the commitment to
6 the enhancing key made in the i 'th public key within
7 PK36 (470). The total signature size is 290 bytes in
8 this example embodiment.

9 Now consider a process for verification of the
10 signature shown in Figure 8. It is desired to verify a
11 purported signature,

12 $S_i = T_i, Y_1, \dots, Y_{23}, Sib(L_i)$

13 on a message m (810) .

14 Compute $h = T2(T_i, H(m))$ (820) .

15 Let h_1, \dots, h_{80} denote the bits of h . Group these
16 bits into 20 groups of 4 consecutive bits. Each group
17 is a number from 0 to 15. Let n_1, \dots, n_{20} denote
18 these numbers (830) .

19 For $1 \leq j \leq 20$, compute $f_j = g^{n_j}(Y_i)$
20 (840) .

21 Compute $N = n_1 + n_2 + \dots + n_{20}$,

1 where N is a number between 0 and 300 and therefore
2 fits in 9 bits. Let n_{21} denote the value of the least
3 significant 4 bits of N, n_{22} denote the value of the
4 next 4 significant bits of N and n_{23} denote the value
5 of the most significant 9'th bit of N (850) .

6 Form $f_{21} = g^{\{15 - n_{21}\}}(Y_{21})$,

7 form $f_{22} = g^{\{15 - n_{22}\}}(Y_{22})$, and

8 form $f_{23} = g^{\{1 - n_{23}\}}(Y_{23})$ (860) .

9 Compute $PK'_i = H(T_i \mid f_1 \mid \dots \mid f_{23})$ (870) .

10 If the signature is correct and the commitment
11 opening string is valid then PK'_i must be the same as
12 PK_i of the key generation phase. To check this
13 compute $L'_i = T2(g_2, PK'_i)$ (880) .

14 If the signature is correct then L'_i must be the
15 same as L_i in the key generation phase. To verify
16 this, since all the siblings of L_i are provided as
17 part of the signature, verify that $M_{\{(i+5)/6\}} =$
18 $T1(g_3, \langle \text{Sib}(L_i), L'_i \rangle)$ where $\langle \text{Sib}(L_i), L'_i \rangle$ is just
19 the concatenation of the L_i 's in the proper order
20 (890) .

00440-50264500

1 Those skilled in the art will recognize that many
2 variations can be built using the approach and/or
3 concepts followed in this example embodiment. The core
4 one-time signature scheme used in this example
5 embodiment will hereafter be referred to as the 80-bit
6 $f(4)$ -scheme, where 80-bit refers to the level of
7 security and output sizes of the underlying
8 cryptographic primitives and $f(4)$ -scheme refers to the
9 application of the one-way function on 4-bit chunks.
10 The example TCR tree construction over 36 instances of
11 the 80-bit $f(4)$ -scheme is herein referred to as a
12 80-bit 36-fanout TCR tree, where 80-bit refers to the
13 size of outputs of intermediate TCR function primitives
14 and the security of the scheme and 36-fanout refers to
15 the number of 1-time public keys that the TCR tree
16 hashes to create a k-time key. The composite scheme is
17 termed a 80-bit 36-fanout $f(4)$ -scheme. From the example
18 description, those skilled in the art can easily create
19 p -bit q -fanout $f(r)$ -schemes for several other values of
20 p , q , and r as the technology and application demands.
21 Increasing/decreasing the value of p will generally
22 result in increasing/decreasing the security of the
23 scheme, with a side-effect on the size of the keys and
24 signatures and the signing speeds. Increasing or
25 decreasing the value of q and/or r will generally
26 create different size/speed tradeoffs while generally
27 maintaining the same level of security.

1 Whereas, the above detailed embodiment describes a
2 signature scheme which includes an embedded TCR
3 commitment, it is to be noted that the embodiment
4 describes only one possible method of creating a TCR
5 commitment scheme. In general, there are multiple ways
6 of creating a TCR commitment scheme from other well
7 known cryptographic primitives. As an illustration of
8 these multiple approaches to realize TCR commitments we
9 further disclose how a TCR commitment scheme can be
10 build from any regular commitment scheme and any TCR
11 function.

12 Now let $H(k,m)$ be any Target Collision Resistant
13 Function taking a key k and a message m , and let C be
14 any regular commitment scheme consisting of a C -commit
15 function as the commitment function, C -decommit as the
16 De-commitment function and C -open as the commitment
17 opening function.

18 Figure 9 shows an example of how the TCR commitment
19 function is built. The data string (901) and the
20 commiter secret information (902), if required by the
21 regular commitment scheme, is first provided as an
22 input to the C -create function (903). The C -create
23 function computes a regular commitment message M as
24 output (904). The regular commitment message M (904)
25 together with a randomly chosen key K (906) is then
26 provided as input to the TCR hash function H (905).

1 Let HKM denote the resulting hash value which is the
2 output of the TCR hash function H applied to the
3 provided key K and the regular commitment message M,
4 that is, HKM equals $H(K,M)$. HKM together with the key
5 K form the TCR commitment message (907). In addition,
6 the commiter, or someone on his behalf may also store
7 the regular commitment message M for to realize
8 additional efficiencies in the computation of the TCR
9 De-commitment function later on.

10 An example of how the TCR De-commitment function is
11 built, is illustrated in Figure 10. First the provided
12 regular commitment message M (904) together with the
13 provided data string (901) and provided secret
14 information (902) is input to the C-decommit function
15 (1001) to yield the regular opening string O as output.
16 The regular opening string O together with the regular
17 commitment message M, form the TCR opening string
18 (1002).

19 An example of how the TCR open function is built is
20 illustrated in Figure 11. First a check is performed
21 (1101) on the provided TCR commitment message HKM, K
22 (907) and the provided TCR opening string O, M (1002).
23 This check consists of computing the function $H(K,M)$
24 and verifying that HKM equals $H(K,M)$. If the check
25 fails then the TCR open function fails (1102) and an
26 error is reported. If the test succeeds then the C-open
27 function (1103) is applied to the provided regular

1 committment message M and the provided regular opening
2 string O to yield the data string (1104) TCR-committed
3 to.

4 The present invention can be realized in hardware,
5 software, or a combination of hardware and software.
6 The present invention can be realized in a centralized
7 fashion in one computer system, or in a distributed
8 fashion where different elements are spread across
9 several interconnected computer systems. Any kind of
10 computer system - or other apparatus adapted for
11 carrying out the methods described herein - is
12 suitable. A typical combination of hardware and
13 software could be a general purpose computer system
14 with a computer program that, when being loaded and
15 executed, controls the computer system such that it
16 carries out the methods described herein. The present
17 invention can also be embedded in a computer program
18 product, which comprises all the features enabling the
19 implementation of the methods described herein, and
20 which - when loaded in a computer system - is able to
21 carry out these methods.

22 Computer program means or computer program in the
23 present context is meant to include any expression, in
24 any language, code or notation, of a set of
25 instructions intended to cause a system having an
26 information processing capability to perform a
27 particular function either directly or after either or

1 both of the following a) conversion to another
2 language, code or notation; b) reproduction in a
3 different material form.

4 It is noted that the foregoing has outlined some of the
5 more pertinent aspects the present invention. This
6 invention may be used for many applications. Thus,
7 although the description is made for particular
8 arrangements and methods, the intent and concept of the
9 invention is suitable and applicable to other
10 arrangements and applications. Thus, it will be clear
11 to those skilled in the art that other modifications to
12 the disclosed embodiments of TCR commitments can be
13 effected without departing from the spirit and scope of
14 the invention. For example, in some cases the
15 TCR-commitment is a regular-commitment. The described
16 embodiments ought to be construed to be merely
17 illustrative of some of the more prominent features and
18 applications of the invention. Other beneficial results
19 can be realized by applying the disclosed invention in
20 a different manner or modifying the invention in ways
21 known to those familiar with the art.